MECE336 Microprocessors I
PIC16F84A

Dr. Kurtuluş Erinç Akdoğan

kurtuluserinc@cankaya.edu.tr

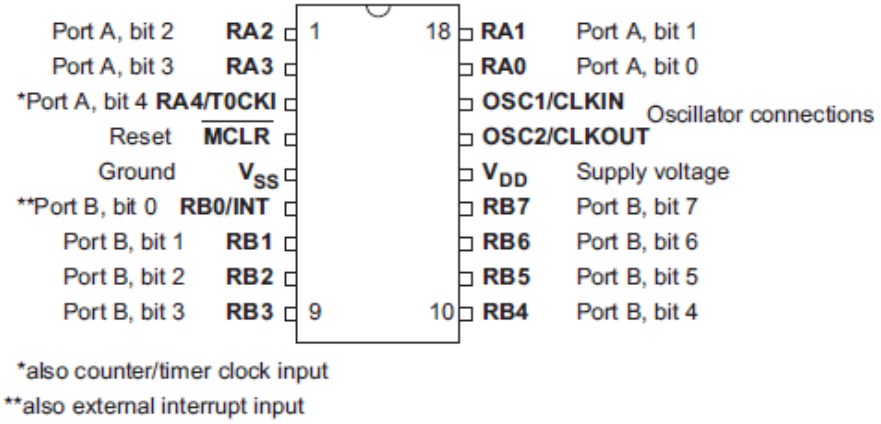
Course Webpage: http://MECE336.cankaya.edu.tr

ÇANKAYA ÜNİVERSİTESİ
MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

CONTENT

- In this lecture we will learn about:
  - The PIC mid-range family, in overview.
  - The overall architecture of the 16F84A.
  - The 16F84A memory system, along with a review of memory technologies.
  - Other hardware features of the 16F84A, including the reset system.

An architecture overview of the 16F84A



-18 pins offers several functions to dedicate

-program address bus is 13-bit allowing 2^{13} (8192) locations in memory to address

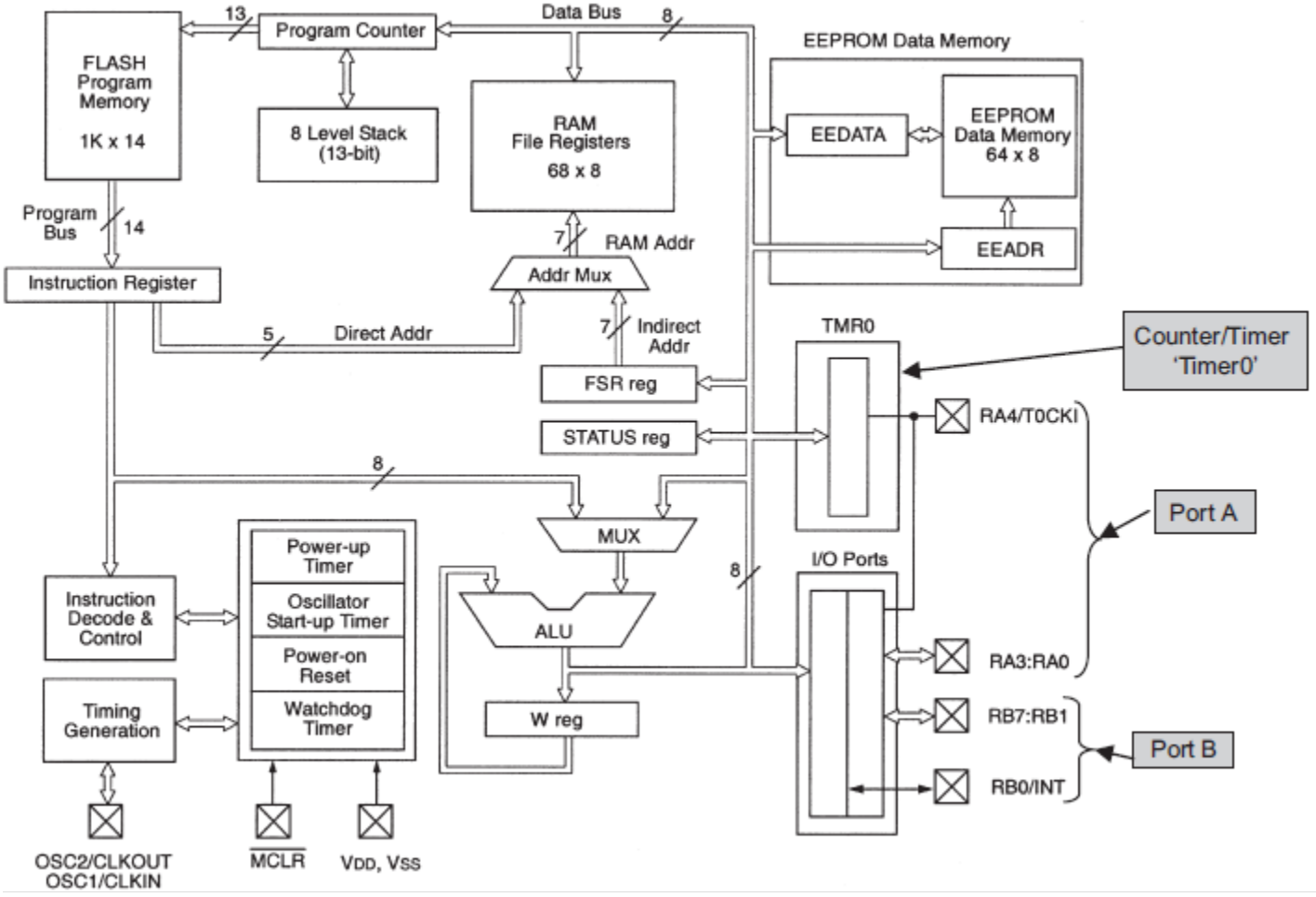
-the instruction word size is 14-bit

-Program memory size, at 1K, is however only one eighth of this.

-EEPROM for program Memory allows programming rapidly and repeatedly changed

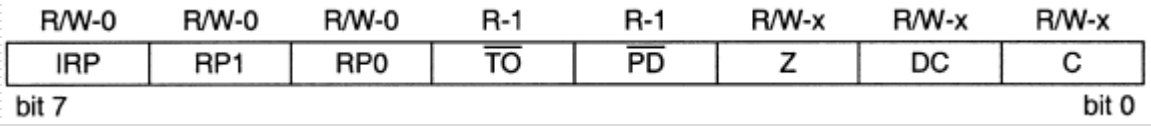
-There are two digital input/output ports, Port A, with five pins, and Port B, with eight.

-Bit 0 of Port B is shared with the external interrupt input.



The Status Register

- The result of any CPU operation is held in the Working register
- What happens when the 8-bit range has been exceeded in an addition instruction, for example by adding binary numbers 1000 0000 and 1111 1101?
- The Working register has no way of indicating this and would simply hold an incorrect result.
- There are flags held in the Status register suc as bits C, DC and Z



R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit Carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

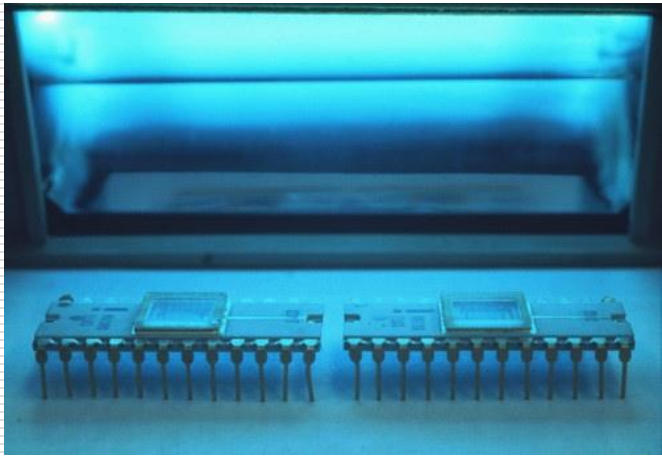
bit 0 **C**: Carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$, the polarity is reversed)
1 = A carry-out from the Most Significant Bit of the result occurred
0 = No carry-out from the Most Significant Bit of the result occurred

A Review Of Memory Technologies

- Static RAM (SRAM) is
  - Volatile
  - mainly used for data memory (RAM) in a microcontroller.
- EPROM (Erasable Programmable Read-Only Memory) technology is
  - non-volatile.
  - erased by exposing it to intense ultraviolet light.
- EEPROM (Electrically Erasable Programmable Read-Only Memory) is
  - non-volatile.
  - erased electrically.
  - written to and erased on a byte-by-byte basis
- Flash is
  - non-volatile.
  - erased electrically.
  - written to and erased on a block-by-block basis
  - Used in latest products suc as USB Drives, solid-state drives

!!!Microchip organized so that a section of memory cells are erased in a single action or «flash».

EPROM

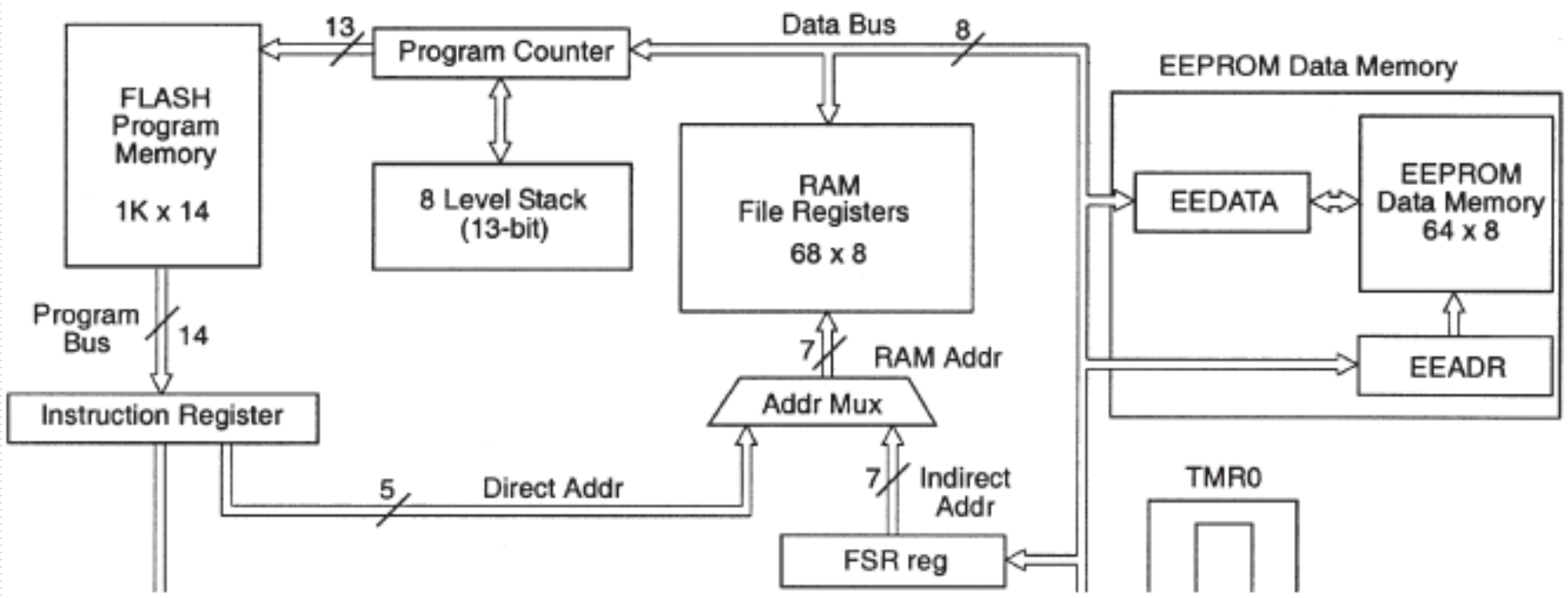


EEPROM vs FLASH MEMORY

EEPROM	Flash Memory
Electrically Erasable Programmable Read Only Memory	Solid State Disk
NOR cells	NAND cells
Less Write More Read	Both Read Write
More Expensive	Less Expensive
Faster Read	Slower Read
More Time For Programming	Less Time For Programming
Less Voltage	More Voltage
Less Erasing Time	More Erasing Time
Erase Byte	Erase Block or Whole chip
Smaller Faster Data	Large Data

PIC Overview: Memory

- Program Memory
  - Flash
  - Program Counter
  - Stack
- Data Memory
  - RAM File registers
  - EEPROM



The 16F84A Memory

- There are no less than four areas of memory in the 16F84A.
- Each memory has its own distinct function and means of access.

Memory function	Technology	Size	Volatile/ non volatile	Special characteristics*
Program	Flash	1K × 14 bits	Non volatile	10 000 erase/write cycles, typically
Data memory (file registers)	SRAM	68 bytes	Volatile	Retains data down to supply voltage of 1.5 V
Data memory (EEPROM)	EEPROM	64 bytes	Non volatile	10 000 000 erase/ write cycles, typically
Stack	SRAM	8 × 13 bits	Volatile	

Program Memory And The Stack

-Program Counter, the Stack and the actual program memory, work together

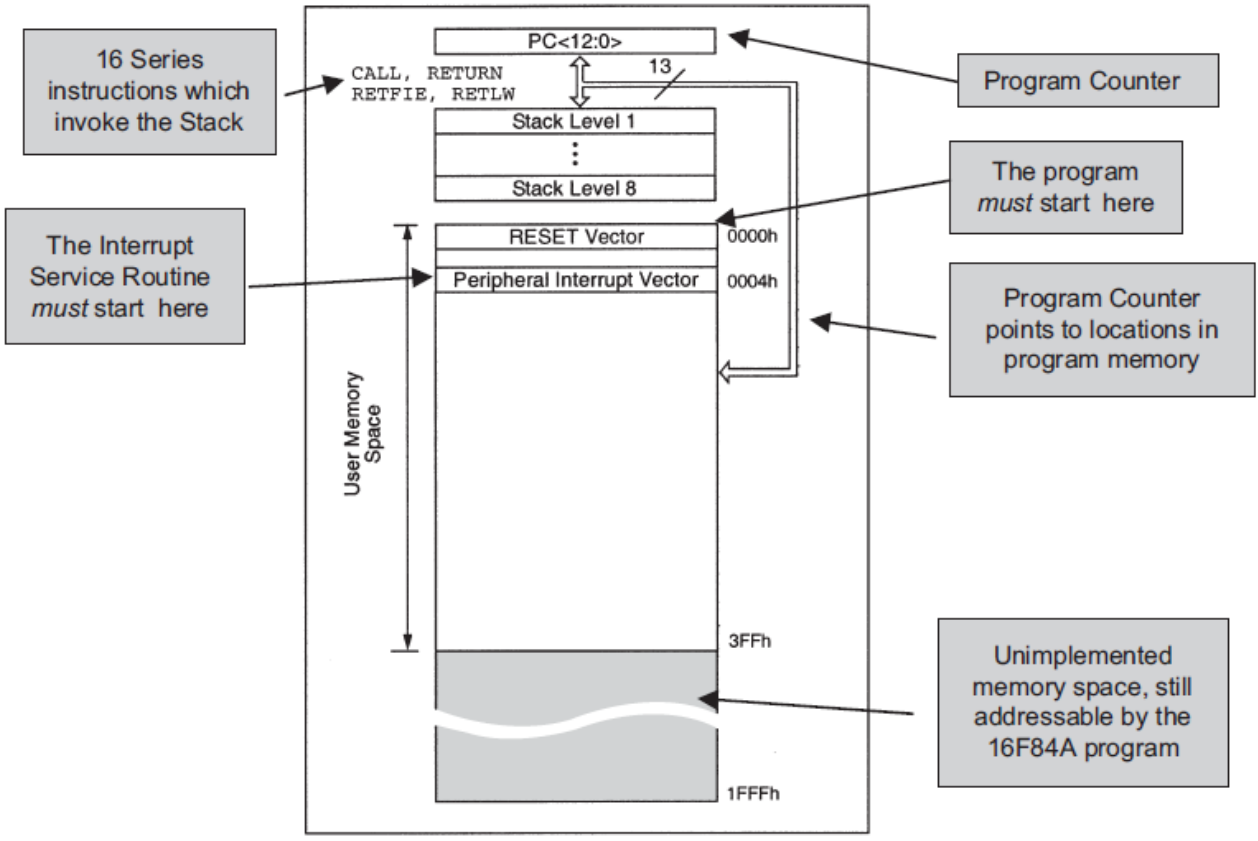
-The program memory is loaded with the program

code that the microcontroller executes.

-The program is in the form of a list of instructions.

-Program Counter acts as a pointer and holds the address of the next instruction that is to be executed by the microcontroller.

-Address range of the program memory is from 0000 to 03FFH. With its 13-bit Program Counter, the microcontroller can theoretically address a range from 0000 to 1FFFH.



Program Memory And The Stack

-Stack is a temporary memory that stores values of the program counter in case of special instructions (CALL, RETURN)

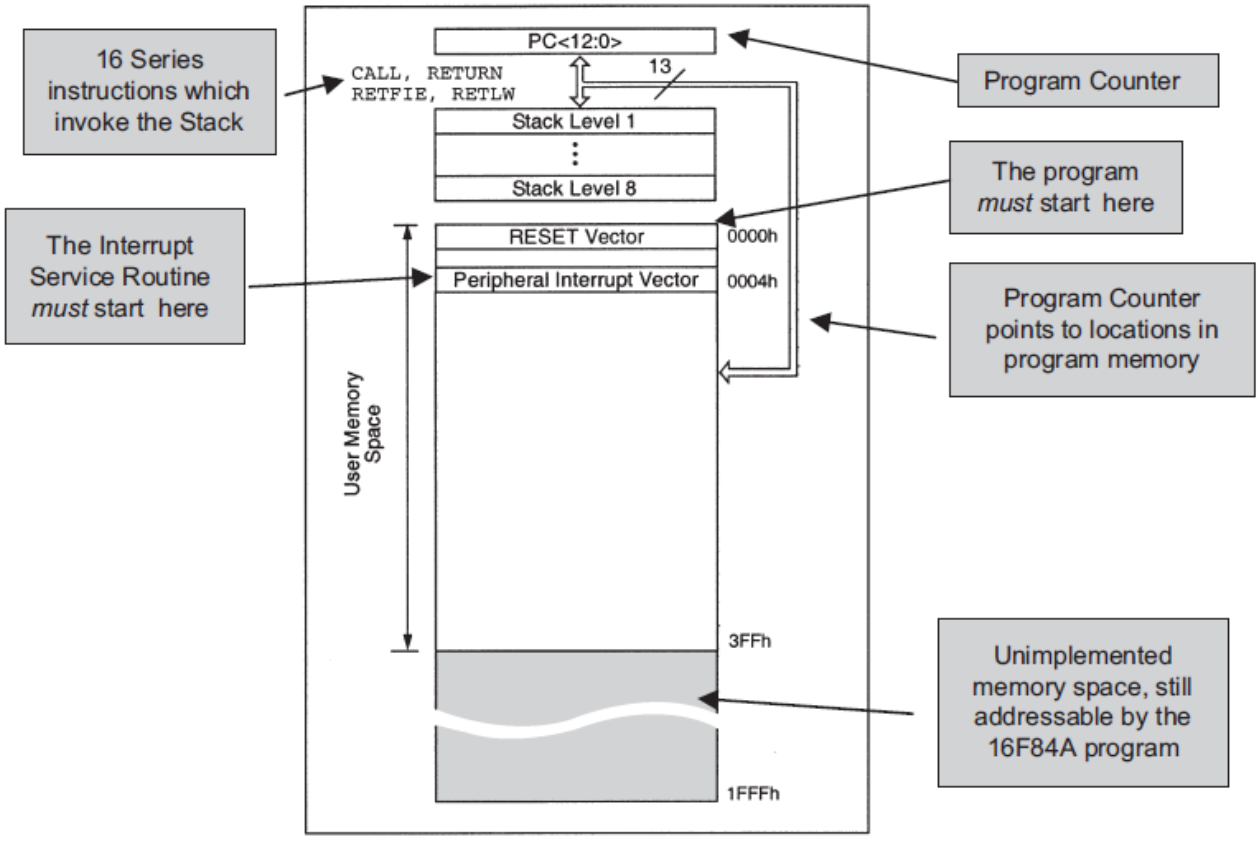
-Stack is structured as LIFO memory – last in, first out

-'reset vector' is first location in the program memory.

-When the program starts running for the first time, for example on power-up, the Program Counter is set to 0000.

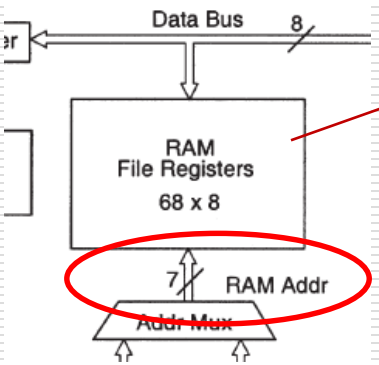
-The programmer must therefore place his/her first instruction at this location.

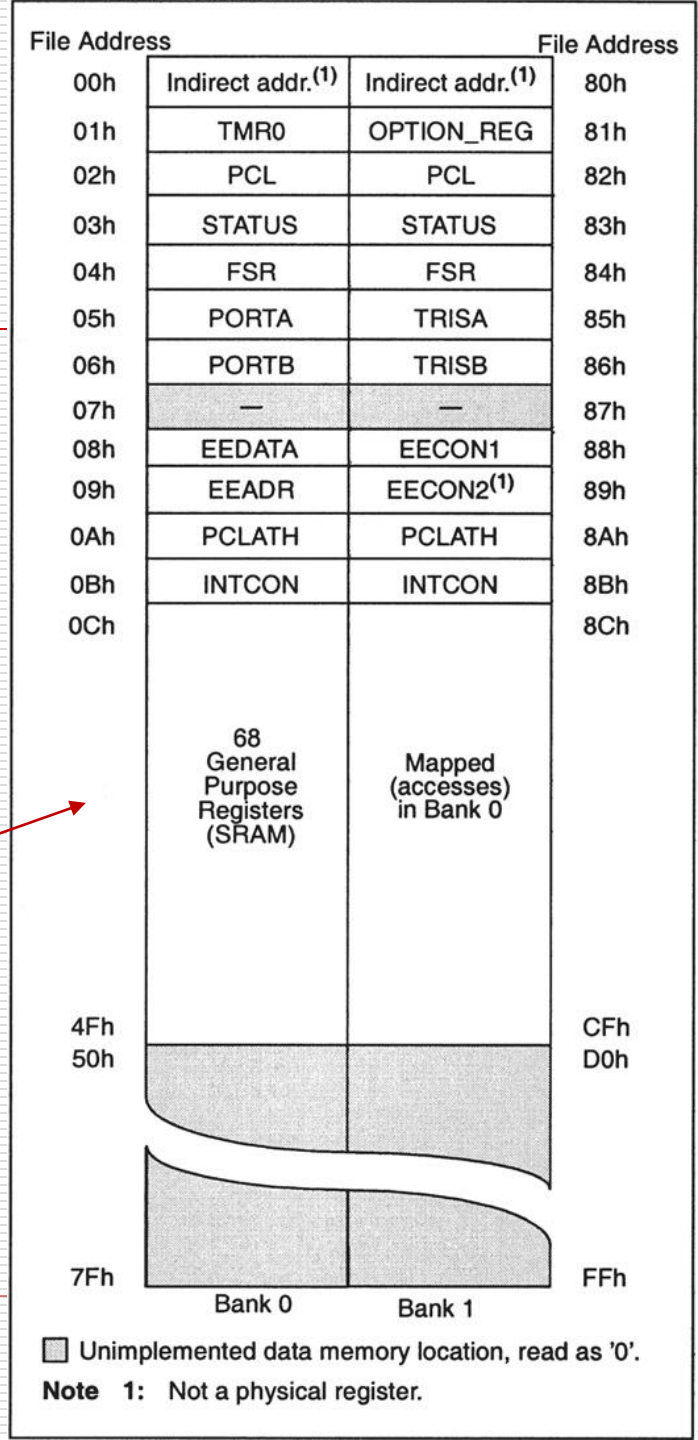
-The 'peripheral interrupt vector' acts in a similar way for interrupt service routines



Data and Special Function Register Memory (RAM)

- The memory area is composed of two banks to use less number of bits for addressing.
  - Bank 0: Address 0x00 to 0x4F
  - Bank 1: Address 0x80 to 0xCF
  - Address 0x50 to 0x7F and 0xD0 to 0xFF are not implemented
  - The address of either bank is the 7-bit RAM address
  - The active bank is selected by bit 5 in the Status register
- First bank is divided into two areas.
  - The first is the general-purpose data memory, which occupies locations $0C_H$ to $4F_H$.
  - Above that are the Special Function registers (SFRs).





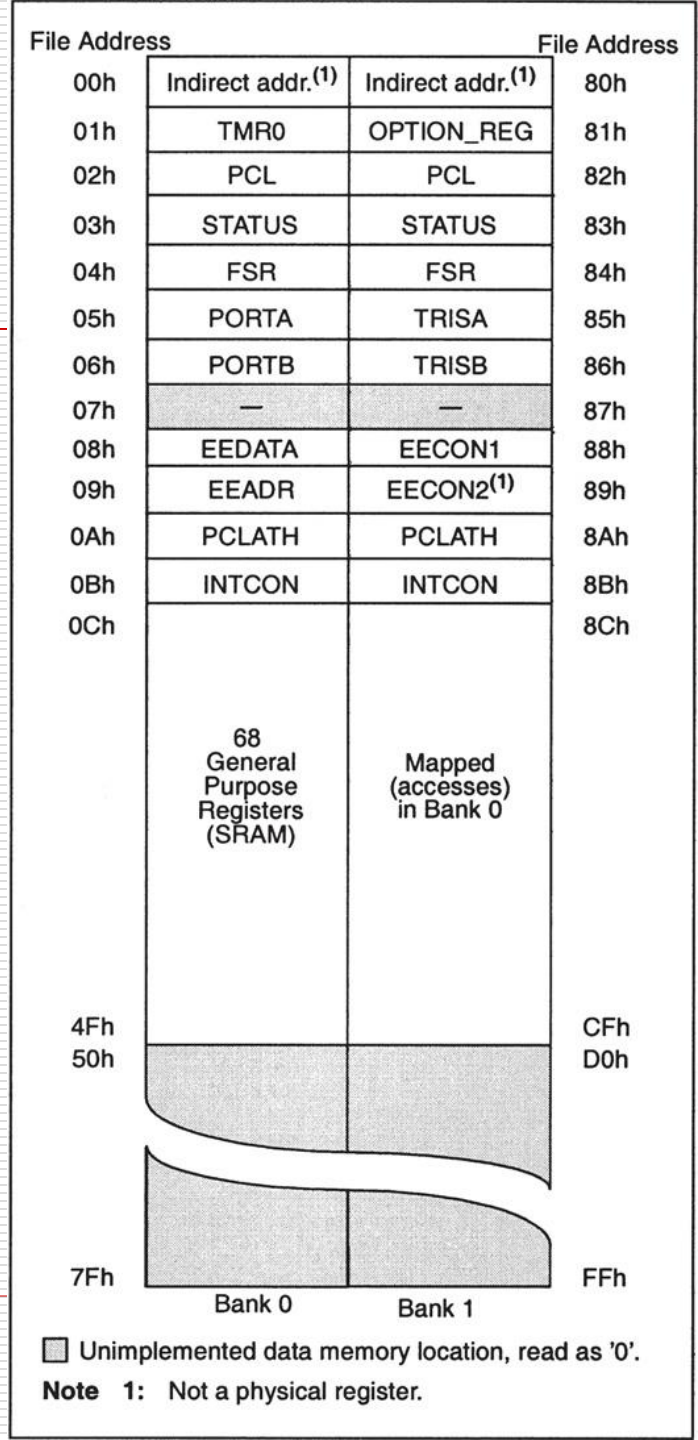
Status register

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 5 **RP0**: Register Bank Select bits (used for direct addressing) 01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

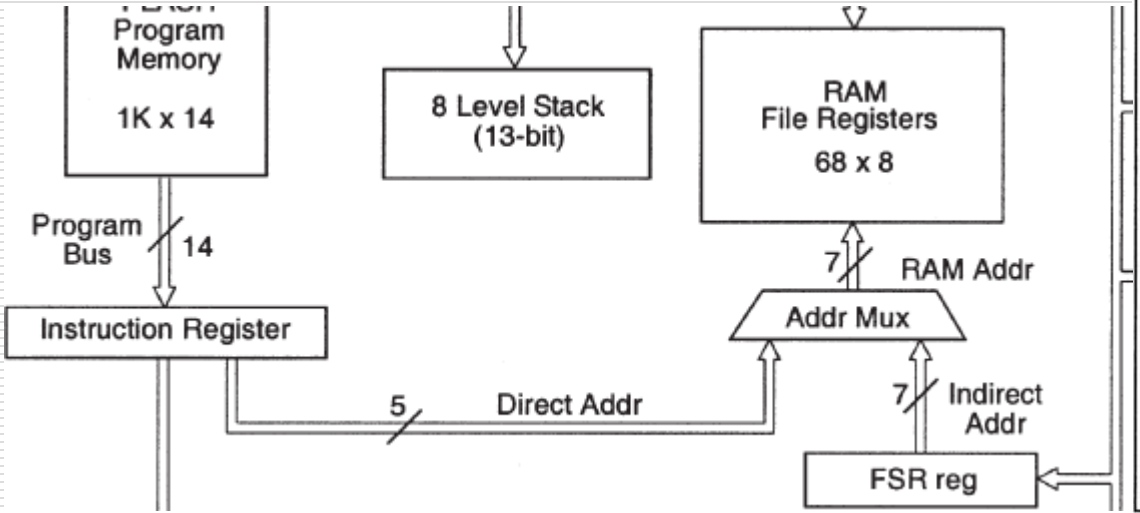
Special Function Registers

- cofigure interaction between CPU and peripherals.
- Each bit is wired across to one or other of the microcontroller peripherals.
- Each is then used either to set up the operating mode of the peripheral or to transfer data between the peripheral and the microcontroller core. As



RAM addressing

- There are two possible sources of the RAM address, selected through the address multiplexer ('Addr Mux').
  - Direct: part of the instruction
  - Indirect: taken from the File Select Register (FSR)
- File address is 8 bit, MSB determines the bank.
  - 7-bit from adress bus from + 1 bit from status registers
  - Or 5 bit from instruction.



File Address			File Address
00h	Indirect addr.(1)	Indirect addr.(1)	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	—	—	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2(1)	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	68 General Purpose Registers (SRAM)	Mapped (accesses) in Bank 0	8Ch
4Fh			CFh
50h			D0h
7Fh			FFh
	Bank 0	Bank 1	

Unimplemented data memory location, read as '0'.

Note 1: Not a physical register.

The Configuration Word

- This allows the user to define certain configurable features of the microcontroller, at the time of program download.
- These are fixed until the next time the microcontroller is programmed.
- it is not accessible within the program or in any way while the program is running.

R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	$\overline{\text{PWRTE}}$	WDTE	F0SC1	F0SC0
bit13													bit0

bit 13-4 **CP:** Code Protection bit
1 = Code protection disabled
0 = All program memory is code protected

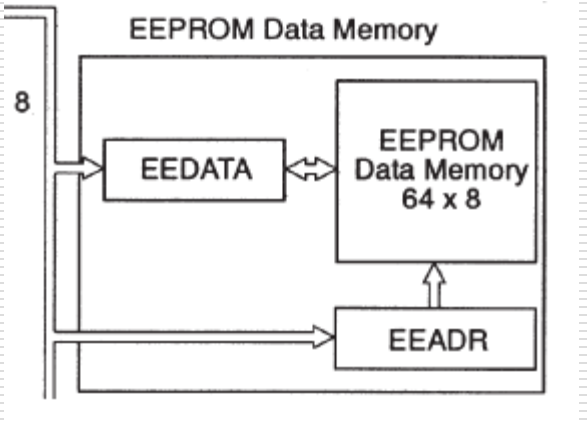
bit 3 **$\overline{\text{PWRTE}}$**: Power-up Timer Enable bit
1 = Power-up Timer is disabled
0 = Power-up Timer is enabled

bit 2 **WDTE**: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **FOSC1:FOSC0**: Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

EEPROM

- The EEPROM is non-volatile and is particularly useful for holding data variables that can be changed but are likely to be needed for the medium to long term.
  - Examples include TV tuner settings, phone numbers stored in a cell phone or calibration settings on a measuring instrument
- In the 16F84A (and indeed any PIC microcontroller), the EEPROM is not placed in the main data memory map. Instead (as the top right of Figure 2.2 neatly shows) it is addressed through the EEADR register and data is transferred through the EEDATA register. These are both SFRs,
- reading from EEPROM is a simple process but writing to it is not



- MPLAB X IDE V. 3.51